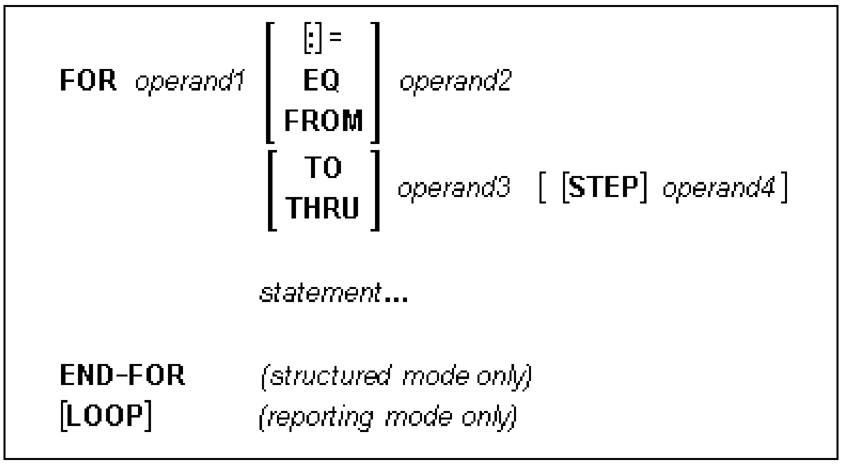
FOR



```
FOR operand1 [ [:]=  ] operand2
             [ EQ    ]
             [ FROM  ]
             [ TO   ] operand3  [ [STEP] operand4 ]
             [ THRU ]

             statement...

END-FOR      (structured mode only)
[LOOP]       (reporting mode only)
```

Operand	Possible Structure					Possible Formats												Referencing Permitted	Dynamic Definition
Operand1		S					N	P	I	F								yes	yes
Operand2	C	S			N		N	P	I	F								yes	no
Operand3	C	S			N		N	P	I	F								yes	no
Operand4	C	S			N		N	P	I	F								yes	no

Related Statement: REPEAT

Function

The FOR statement is used to initiate a processing loop and to control the number of times the loop is processed.

Control Variable - *operand1* and Initial Setting - *operand2*

Operand1 is used to control the number of times the processing loop is to be executed. It may be a database field or a user-defined variable. The value specified after the keyword FROM (*operand2*) is assigned to the control variable field before the processing loop is entered for the first time. This value is incremented (or decremented if the STEP value is negative) using the value specified after the STEP keyword (*operand4*) each additional time the loop is processed.

The control variable value may be referenced during the execution of the processing loop and will contain the current value of the control variable.

TO Value - *operand3*

The processing loop is terminated when operand1 is greater than (or less than if the initial value of the STEP value was negative) the value specified for operand3.

STEP Value - *operand4*

The STEP value may be positive or negative. If a STEP value is not specified, an increment of "+1" is used.

The compare operation will be adjusted to "less than" or "greater than" depending on the sign of the STEP value when the loop is entered for the first time.

Operand4 must not be "0".

Consistency Check

Before the FOR loop is entered, the values of the operands are checked to ensure that they are consistent (that is, the value of *operand3* can be reached or exceeded by repeatedly adding *operand4* to *operand2*). If the values are not consistent, the FOR loop is not entered (however, no error message is output).

Example

```
/* EXAMPLE 'FOREX1S': FOR (STRUCTURED MODE)
/*****************************************
DEFINE DATA LOCAL
1 #INDEX (I1)
1 #ROOT  (N2.7)
END-DEFINE
/*****************************************
FOR #INDEX 1 TO 5
 COMPUTE #ROOT = SQRT (#INDEX)
 WRITE NOTITLE '=' #INDEX 3X '=' #ROOT
END-FOR
/*****************************************
SKIP 1
FOR #INDEX 1 TO 5 STEP 2
 COMPUTE #ROOT = SQRT (#INDEX)
 WRITE '=' #INDEX 3X '=' #ROOT
END-FOR
/*****************************************
END
```

```
#INDEX:    1   #ROOT:   1.0000000
  #INDEX:    2   #ROOT:   1.4142135
  #INDEX:    3   #ROOT:   1.7320508
  #INDEX:    4   #ROOT:   2.0000000
  #INDEX:    5   #ROOT:   2.2360679

  #INDEX:    1   #ROOT:   1.0000000
  #INDEX:    3   #ROOT:   1.7320508
  #INDEX:    5   #ROOT:   2.2360679
```

Equivalent reporting-mode example: See the program FOREX1R in the library SYSEXRM.